Distributed Systems

Lec. slides 1

↝ Centralizing of activities is not the natural way of handling way, opposite of distribution.

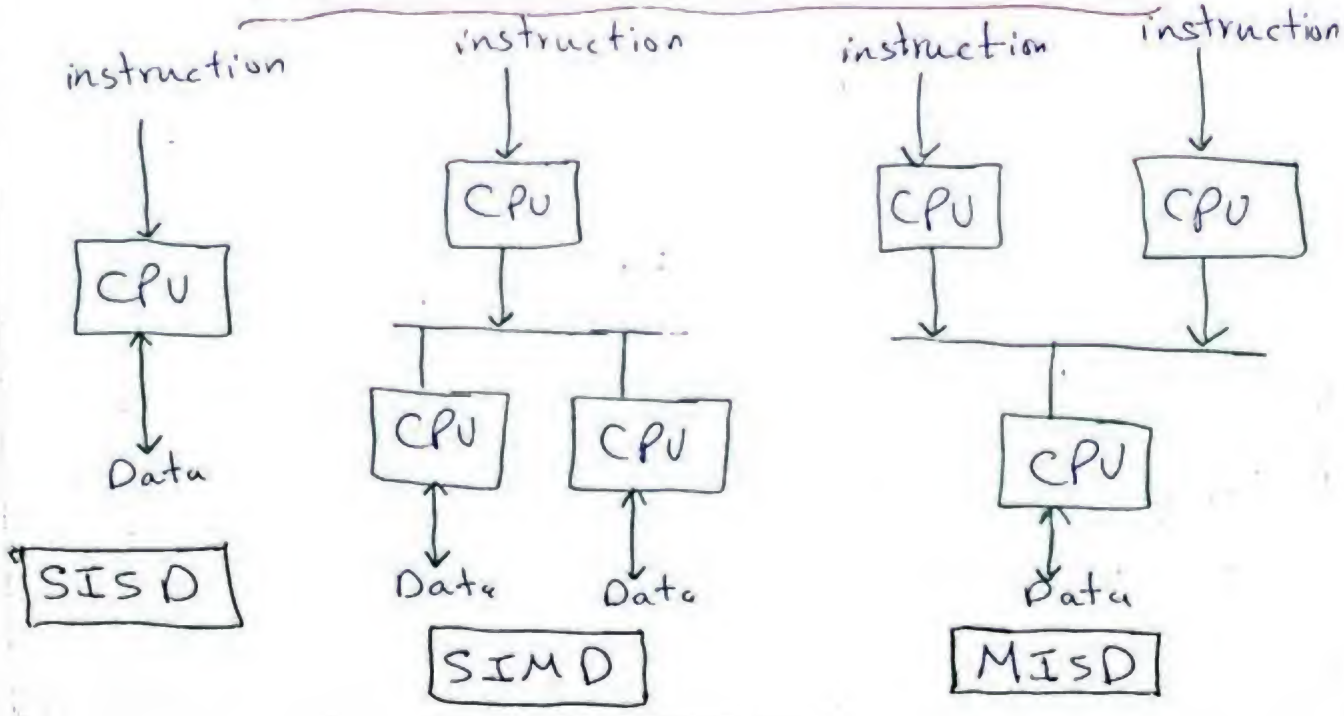
Definition of distributed system

1) Collection of independent Computers that appear to the users of the system as a single computer.

2) A group of connected components that are cooperating to perform a single task (in parallel)

3) InterConnected devices that are sharing data and resources.

Flynn's taxonomy for Computer's HW



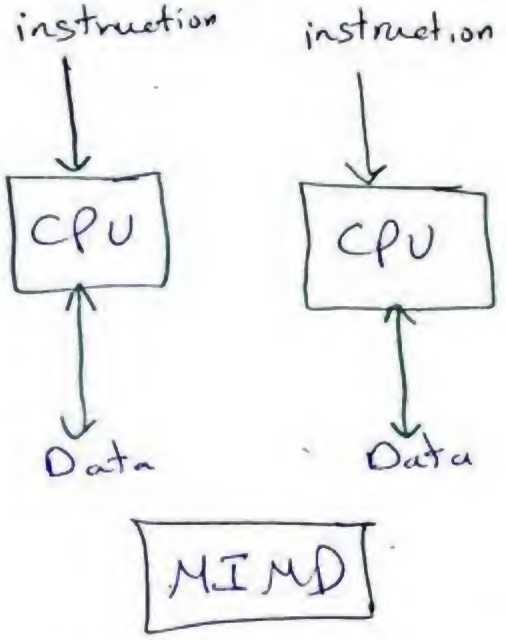
Where:

1) SISD (ordinary PC)
↳ single instruction single data.

2) SIMD (Array Processor)
↳ single instruction Multiple data

3) MISD (Pipelined Processors)
↳ Multiple inst. single data.

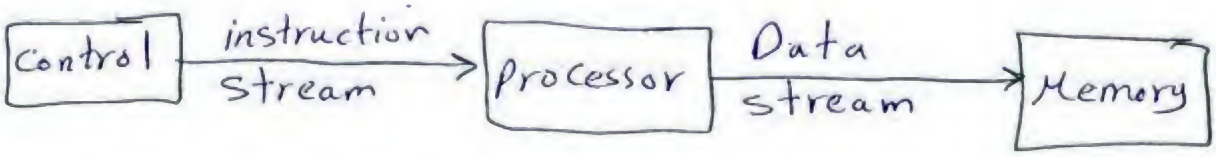
4) MIMD (Multiprocessors or multicomputers)
↳ multiple inst. Multiple data.



---

1) SISD

→ Called von Neumann Computers.
→ starts from Personal Computers to large mainframes.
→ sequential machine Process one instruction at a time on single data in single Processing element.
→ speed limited by CPU speed and bus speed.



*To speed up, add Concurrency by:

1) Concurrent execution of several different user's Programs ... ...

more than Program share CPU:

a) one uses CPu for a while & when it releases the CPu another Program Can use it

2) eXecute I/o oPerations simultaneously with the exection of user's Program.

3) using Parallel functional units inside the Processing unit to be multifunction, so as more than one unit Can function at the same time.

4) Pipelining where instruction is broKen to its element Parts & each Part assigned to a ProCessing element

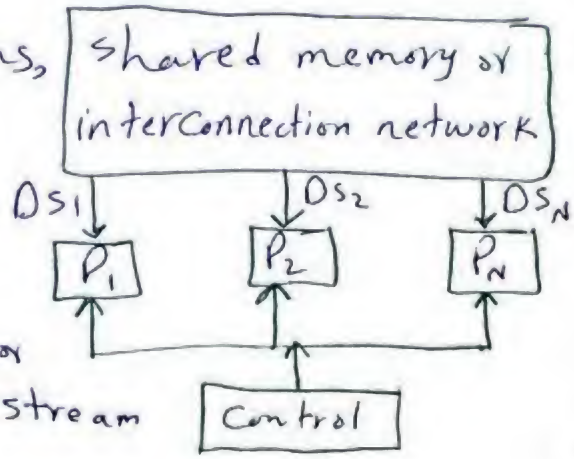
---

SIMP (single instruction multiple data)

↳ All N identical Processors oPerate under Control of single instruction issued by Central control unit.

→ Processors oPerate Synchronously.

→ there are N data streams, one Processor so different data Can be used in each Processor.



P → Processor
DS → data stream

what is the role of Global clock

↳ used to ensure lockstep operation.

i.e ⇒ at each step (it ticks) all Processors execute the same instruction, each one different data stream

⇒ SIMD is best used in apps. that contain operation on matrices or vectors.

→ each Processor will execute the same operation but on different dataset which is part of matrix

- total result stored in shared memory.

- Processing elements (PEs) are slaves under control of master CU.

* → Functions of CU

a) decoding instruction.

b) generate micro-operations (needed for execution for the PEs)

c) generate and broadcast Common memory addresses.

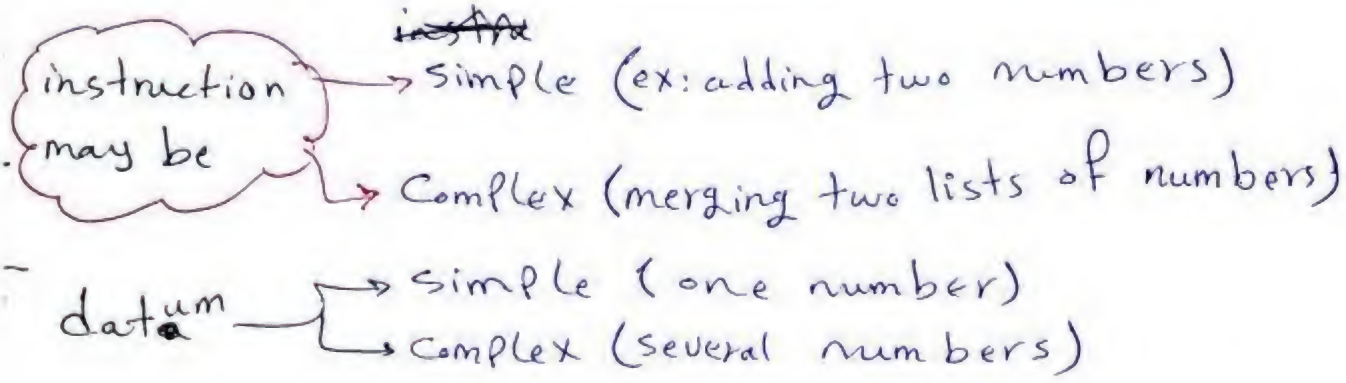
d) receive the data from PEs and Producing final results.

Ex Adding two matrices $A + B = C$

assume we have A, B → two matrices of order 2 and 4 Processors.

* $A_{11} + B_{11} = C_{11}$ ... $A_{12} + B_{12} = C_{12}$
* $A_{21} + B_{21} = C_{21}$ ... $A_{22} + B_{22} = C_{22}$

→ Same instruction is issued to all 4 Processors and all Processors execute the instructions simultaneously.



⇒ information Can be encoded in the instruction itself indicating whether:

- Processor is active (execute the instruction)
- Processor is inactive (wait for next instruction)

→ in solving Problems by SIMD and MIMD, Computers It is useful for Processors to be able to Communicate with each other to exchange data or results,

How could Processors communicate with each other to exchange data?

a) using shared memory and shared variables.

b) using some form of interconnection network and message passing (distributed memory)

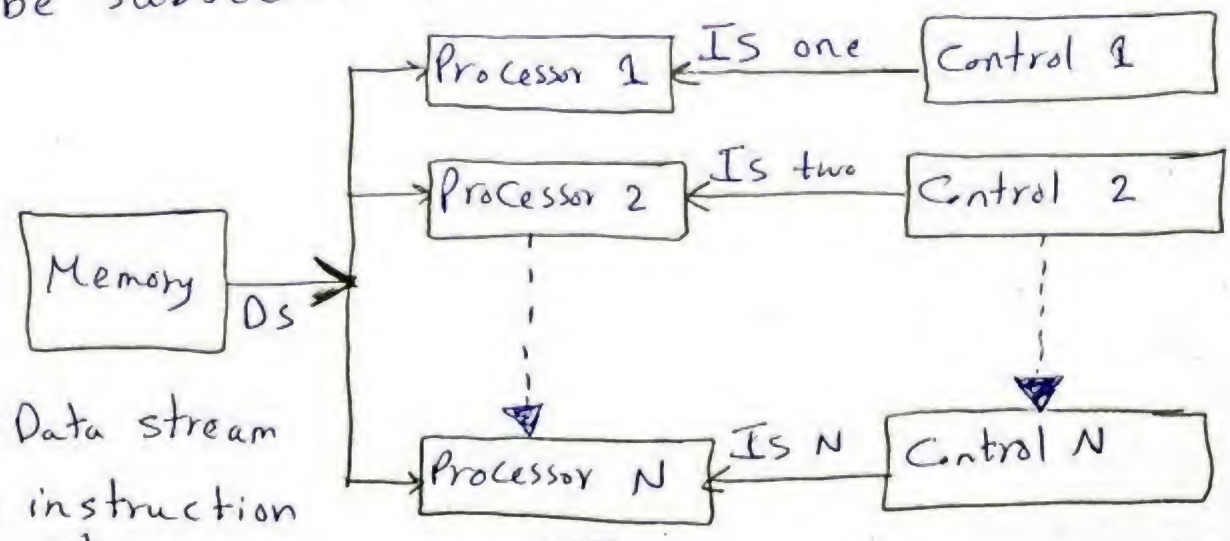
SDMI (single data multiple instruction)

→ N Processors, each with its own control unit share a common memory.

→ N streams of ~~d~~ instructions & one stream of data.

* How Parallelism is achieved?
  ↳ by letting Processors do different things at the same time on the same datum.

* When did MISD machines are useful?
  ↳ in computations where the same input is to be subjected to several different operations.



Ds = Data stream

Is = instruction stream

MISD (multiple instruction single data)

For example check whether Z (number) is Prime
↳ simple solution (try all possible divisions of Z)
Assume no. of Processors (N) is given by (Z-2 = N) (All Processors take Z as input and tries to divide it by its associated divisor. So
it is possible to check if Z is Prime (in one step)
More realistically if N < Z-2 then a subset of divisors would be assigned to each Processor.

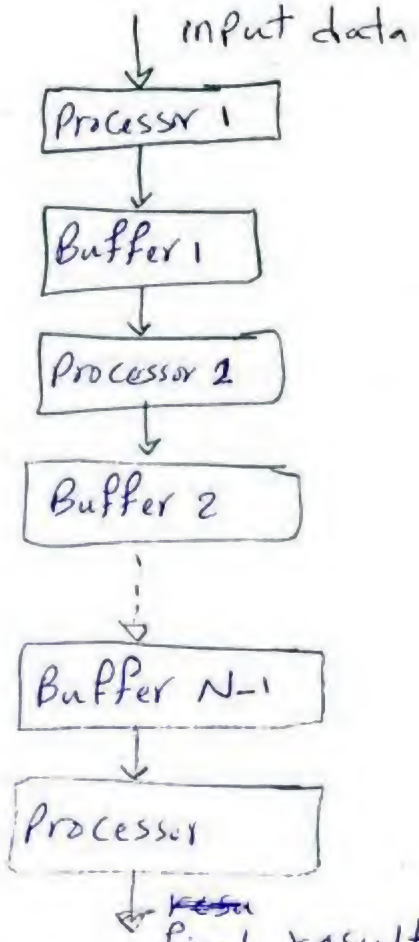
⇒ for most applications MISD is no commercial machines exist with this design.

→ Pipelined Processors is example of MISD.

Pipelined Processor

→ machine that overlaps computations by subdividing and interleaving the operation of these sub-computations each on individual Processor.

→ basic idea break Complex time Consuming function into series of simpler sequential operations.



* o/p of one stage is the i/p of the next.

* use of Buffers

hold output data temporally unit the next stage finishes its data execution.

$$\text{speed up} = \frac{NM}{N+M-1}$$

speed up of Pipelined is

↳ no. of times faster a pipeline component that operates than a non pipelined component that performs the same operation.

where) N → no. of data streams.

M → no. of stages (each stage require 1 time unit)

NM → total time taken to complete N operations in non pipelined component.

Perfect usage of pipelined idea ⇒ time should be the same for all stages.

* if one stage takes less time, the remaining time will be wasted.

MIMD (multiple instruction multiple data)

→ most general & most powerful.

→ N Processors, N streams of instruction and N streams of data.

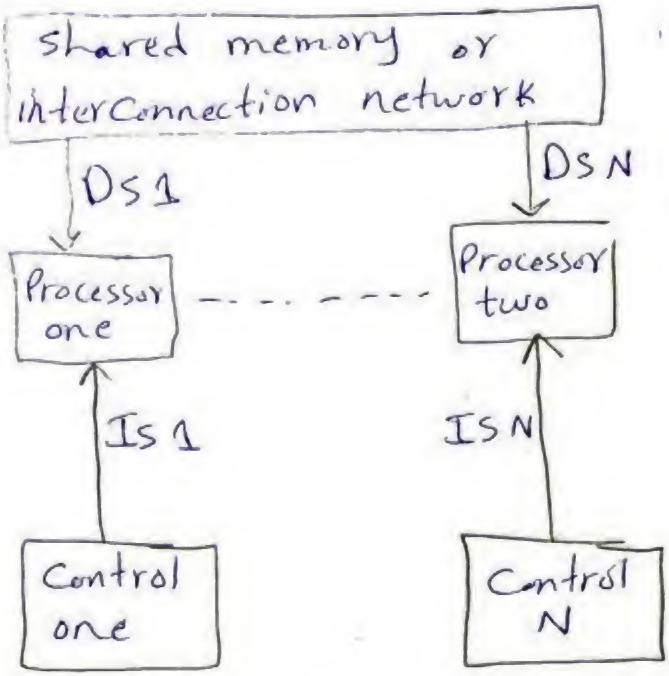
→ each Processor is fully Programmable and Can execute its own Program.

→ It operate under control of instruction stream issued by its own control unit.

→ Processors operate asynchronously, Can do different things on different data at the same time.

* MIMD Computers with shared memory are Known as multiProcessors or tightly Coupled machines.

For examples ENCORE, MULTIMAX, SEQUENT & BALANCE.



* MIMD Computers with interConnection network Known as multiComputers or loosely Coupled machines

ex INTEL iPSC, NCUBE/7 and transputer networks.

Tightly and loosely Coupled Systems

→ Another dimenson of our taxonomy.

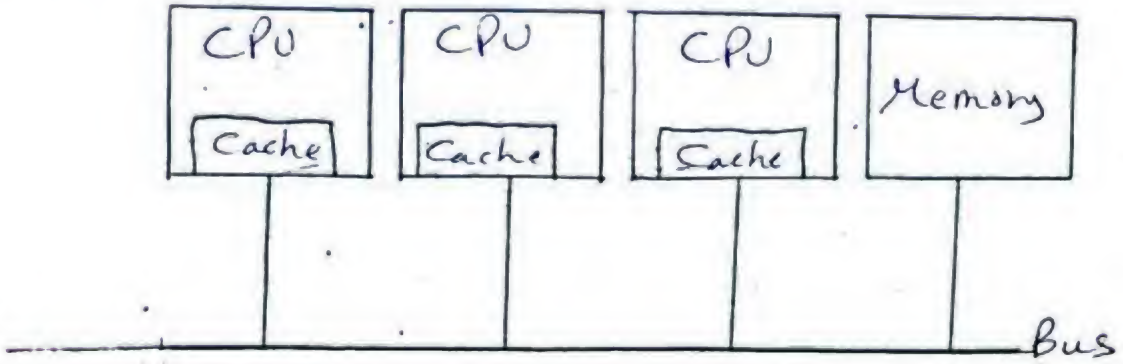
In tightly Coupled System

→ delay experienced when the sent message between two computers is small and data is high (no. of bits per second is large)

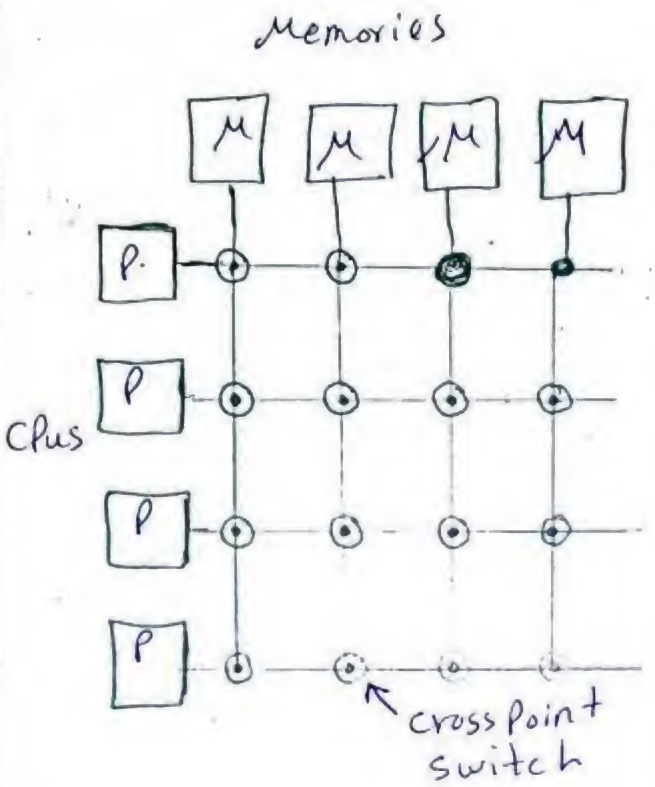
→ It is referred to **multiProcessors** when no. of Processors each: ~~having its memory and are organized on single chassis~~.

1) having its memory.
2) organized on single chassis.
3) have access to common memory through high speed bus under control of single OS.

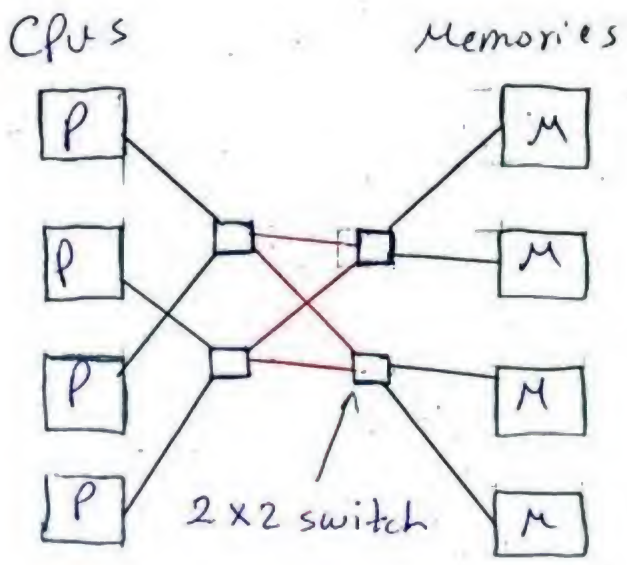
Ex hyper cube and tree network.



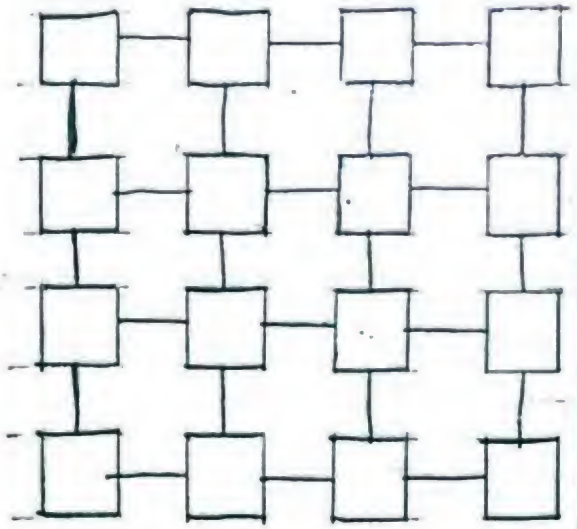
A bus-based multiprocessor.



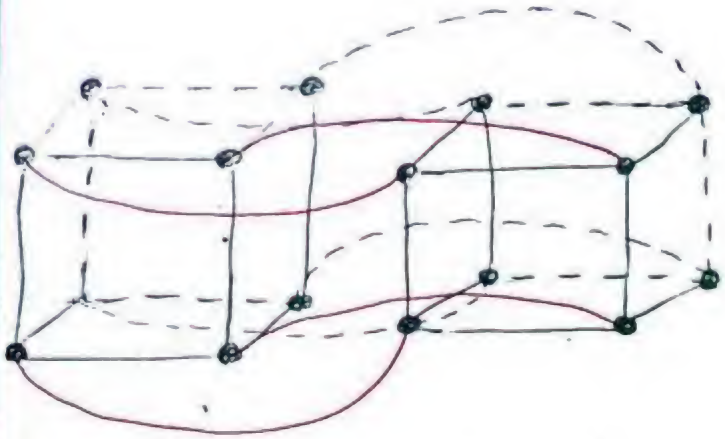
"Crossbar switch"



* An omega switching network



Grid



Hypercube multiprocessor

Grid

a) rectangular array of CPUs with connections only to neighbours.

b) messages from one corner to opposite corner in an (n by n) grid $(n-1)+(n-1)$ hops.

Hypercube

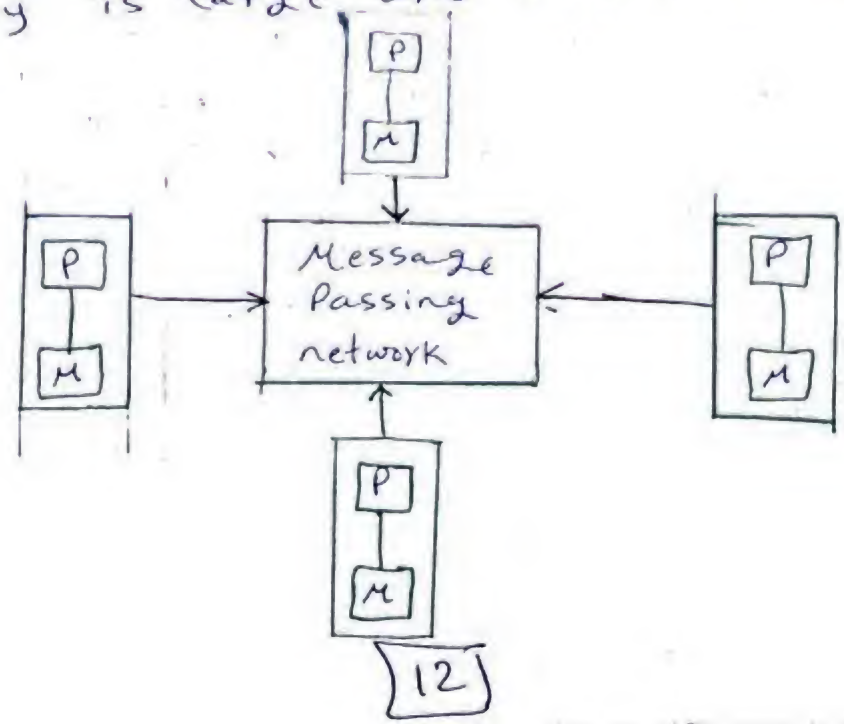
a) n-dimensional cube with CPUs at the vertices.

b) there are $2n$ CPUs.

c) messages from one CPU to CPU most distant from it require n hops $[= \log_2 (\text{nr. of CPUs})]$.

Loosely-Coupled system

→ there are multiple computers that communicate at the I/o level through intermachine message, so delay is large and data rate is low.



→ to complete a transfer of data between two processors, ~~they~~ cooperation between them is required at OS level.

tightly Coupled vs loosely Coupled

1) For example, two CPU chips on the same printed circuit board and connected by wires etched onto the board are likely to be tightly Coupled ~~whereas two~~
→ two computers connected by 2400 bit/sec modem over the telephone system are certain to be loosely coupled.

2) tightly coupled systems tend to be used more as parallel systems (work on single problem) ; loosely Coupled tend to be used as distributed system (work on many unrelated Problems), this not always true.

3) It is possible to have system that is Physically loosely Coupled ~~and~~ but logically tightly Coupled.

4) Grid and clusters are examples of loosely Cupled that can work ~~4~~ in orderly fashion to accomplish one task

Table of Computer Generations in 50 years
↳ Please look at Page 27 at slide 1.

The need for high performance Computers

1) many of today's applications are very Computationally intensive and require vast amount of Processing Power.

2) if several operations can be Performed simultaneously then total Computation time is reduced.

Note
↳ distributed system version has the Potential of being 3 times as fast as the sequential machine.

Defining speedup and efficiency

*Parallel algorithm : is an algorithm for execution of Program which involves the running of two or more Processes in two or more Processors simultaneously.

* Speedup an efficiency are two important measures of quality of Parallel algorithm.

Speed up = $SN = Ts / Tp$

Efficiency = $EN = SN / N$

Where

$Ts \rightarrow$ time taken to run the fastest serial Algorithm on one Processor.

$Tp \rightarrow$ time taken by Parallel Algorithm on N Processors.

Linear speed up → Produce speedup of N using N Processor and efficiency of 1 (100 %)

Factors that limit speed up

1) Software overhead

↳ It arises in the concurrent implementation (eg. there may be additional index calculations necessitated by the manner in which data are "split up" among Processors.

Parallel Program → more lines of code executed

Sequential → less lines....

2) Load Balancing

↳ Generally speedup is limited by speed of slowest node, so an important consideration is to ensure that node performs same amount of work (system is load balanced)

3) Communication overhead

↳ if Communication & calculation Cannot be overlapped.

So any time spent Communicating data between Processors degrades speedup (Cause processors are not calculating)

Distributed systems vs centralized systems

1) non-functional requirements and global nature of the apps. makes us decide which of them to build.

2) distributed system Can be defined as collection of autonomous hosts that are connected through a computer network.

→ host is computer executes components that form part of Distributed system.

Advantages of Distributed systems

1) Performance:

a) collection of Processors Can provide higher Performance than Centralized Computer.

b) distributed sys. may have computing power than centralized mainframe.

2) distribution

↳ many apps. involve by their nature, spatially separated machines.

3) Reliability (fault tolerance)

↳ if some of machines crash, system can survive.

4) Incremental growth

↳ as requirements on processing power grow, new machines can be added incrementally.

5) sharing of data/resources

↳ shared data is essential to many apps. (banking) and outher resources can be shared.

6) Communication

↳ makes human-to-human communication easier.

Disadvantages of distributed systems

1) Difficulties of developing distributed software

↳ imagine shape of OSs, apps, programming languages.

2) Network Problems

↳ It can arises from loss of messages, overloading...

3) security Problems

↳ sharing generates problem of data security.

Charcteristics of Distributed system

1) Resource sharing

→ Resource is provided by Computer which is member of distributed sys. Can be shared by clients and other members of system on network.

* Resource manager → software module that provides interfaces which enables resource to be manipulated by clients.

2) openness (extensible in various ways)

→ new resource sharing services can be incorporated without disruption or duplication of existing services.
→ every service is equally accessible to every client.
→ It is easy to implement, install and debug new services.
→ user can write and install their own services.

3) Transparency

*hide all unnecessary details from users.
*clients don't need to know location of servers.

4) share ability

↳ Allow the comprising systems to use each other's resources.

5) Expandability

↳ Permits new systems to be added as members of overall system.

6) local Autonomy

↳ each Processor is able to manage its local resources.

7) Improved Performance

* factors that influence on Performance of distributed sys.

a. Performance of individual work stations.

b. speed of Communication infrastructure.

c. extent to which realiability is Provided.

d. flexibility in workload allocation.

8) Improved reliability and availability

↳ Disruption would not stop whole system from providing its service ~~at~~ as a resources spread across multi computers.

9) Concurrency

↳ All concurrent access must be synchronized to avoid problems such as lost update and dirty read.

10) Scalability

→ system should remain efficient even with increasing in no. of users and connected resources.

→ Cost of adding resources should be reasonable.

→ Performance loss with increased no. of users and resources should be controlled.

11) Heterogeneity (متجانس غير)

→ not having same things like Programming languages, OSs, HW platforms.

→ Distributed apps. are typically Heterogeneity.

⇒ to solve this using Middleware (SW layer) to mask Heterogeneity.

12 Fault tolerance

→ system handles any error occured.

→ " continues to work even if parts of it fails.

→ Accomplished via HW redundancy, SW recovery.

some definition on fault tolerance

a) fault tolerance

↳ related to reliability, system has to detect faults and act in reasonable way.

b) mask the fault: continue to work with possibly reduced performance but without loss of data.

c) fail gracefully ~~react~~ to predict the fault and stop functionality for short period, but without loss of data/info.

Notes

→ Data or system musn't be lost and copies must be ~~constant~~ consistent.

→ The more copies kept, the better the availability but keeping consistency becomes difficult.

Transparency

* How to achieve single system image?
* How to fool everyone into thinking that collection of machines is simple computer?

a) Access transparency

↳ local and remote resources are accessed using identical operations.

b) location transparency

→ users cannot tell where HW and SW resources are located.

c) Migration (mobility) transparency

↳ resources should be free to move from location to another without change names.

d) Replication transparency

↳ system is free to make additional copies of files and other resources, without users noticing.

e) Concurrency transparency

↳ users will not notice existence of other users in system.

f) Failure transparency

↳ apps. should be able to complete their tasks despite failures occuring.

e) Performance transparency

→ load variation should not degrade performance.

→ This achieved by automatic reconfiguration as response to changes of load.

↳ difficult to achieve.